BUILD TIME DETERMINATION AND INSTALLATION OF DRIVERS ON CLONED SYSTEMS

FIELD OF THE INVENTION:

[0001] The present invention is related to systems, program products and methods for creating replica computer system program images, more particularly to providing a replica computer system program image then determining and retrieving appropriate program components.

BACKGROUND OF THE INVENTION:

[0002] Organizations have responsibility for supporting diverse computers and work stations. A medium sized organization may have thousands of diverse models and types of computers that need to be managed and maintained to appropriate levels or versions of applications, operating systems and support hardware (drivers). The organization needs to be concerned with availability, serviceability, new installations, maintenance and security.

[0003] When adding a computer or workstation, organizations often supply a standard computer system "image" of software (including an Operating System) to the new computer. The image must be customized according to machine type, model and peripheral attachments. The image may also be customized to provide a set of applications and hardware drivers. This allows the organization to maintain a reduced set of images for all of the computers that need to be supported. Even with this technique, the number of images can be large and each image may be large as easily can be seen by the fact that an image must include all of the driver programs that a computer or workstation may need. Managing a large number of versions is a logistics problem but having large numbers of large images creates a need for large storage. Distributing the large images by way of a network is time consuming especially when the network includes a slow protocol or when there is high demand for the network.

[0004] Native installation of operating systems and applications can be time consuming and error prone. Realizing this, companies such as PowerQuest and Symantec have developed disk imaging software that takes a "snap shot" of a computer system image and saves it to a file(s). The "snap shots" can be saved on a server and distributed to new clients by way of a network. The saved "snap shot" is a "clone" or "replica" image of the computer system image. The "snap shot" can then be used by other computers by loading the "snap shot" into another identical computer (cloning).

[0005] Cloning of a computer image is the process of configuring a master computer system with a specific operating system and set of application programs supporting a specific set of peripheral devices and copying (cloning) the "program image" of this configuration to other (cloned) computer systems. The copied image is a clone of the master computer system image.

[0006] Thus, all of the systems cloned have common software (with the same levels and versions of the same software). Cloning simplifies an organization's IT infrastructure since all of the computers or workstations have the same software. A problem exists with this strategy since a separate clone image must be created to support different computer types and models, hardware configuration differences, device drivers and the like.

[0007] Because of its ease of use and fast speed, cloning has solidified its place in the industry by allowing companies to simplify client deployment and reduce costs by putting a client standardization effort in place.

[0008] Referring to Fig. 3, an example cloning process is shown. A client 1 301 computer system is configured to represent the desired clone system. It comprises a characteristic machine type/model (TX1) 305, a desired operating system (OS) 302, optional applications 303, desired drivers 306 and desired configuration information 304. The image of client 1 is then captured and stored as Image2 318 in a server 316 storage area. The Server 316 holds N images. Client 2 307 is to be built. Since the machine type is TB2, desired image2 is deemed appropriate so it is loaded into the client2 307 via a network connection 321 or CD. Thus, client 2 307 becomes a clone of client1 301. The server may support cloning of other computers 312, each one having to select the appropriate image stored in the server 316.

[0009] Current cloning software does not address the issue of adding new hardware, driver, and application support to an existing image file. Since replicating system images (cloning) requires that the target system 307 and the source system 301 be exactly the same, large enterprises can sometimes have hundreds or even thousands of cloned images 317-219 on their servers 316 in order to have a cloned image for each supported system configuration. Some operating systems, like windows 2000/XP from MICROSOFT CORP., have imaging tools that allow a technician to reset the devices that had been previously detected by the operating system. By resetting the devices, the operating system can re-scan for hardware on the next boot of the operating system. Using this method, in a cloned system, requires managing the plug and play driver repository on the cloned system. In order to create a new clone image 318, the technician would clone a known image file 317 containing an operating system with installed and customized applications to a system, boot the operating system, manipulate the driver repository 306 that resides on the client 301 to include new or updated drivers, run the imaging tool, then clone the system image back to the server 316. When the technician saves the "cloned client system" to the server, the technician is saving a snapshot of the operating system 302, applications 303, and hardware support components 304. The snapshot is a version of the computers' operating system that can be cloned (replicated) to other systems that are supported by the updated driver repository 306 or native driver support built into the operating system.

[0010] This method isn't without its problems. Plug and play systems force the image to use the latest available driver in the repository where the organization would prefer to have a predetermined version of the driver. Also, updating and adding drivers to the master image file can be tedious, time consuming, and error prone.

MICROSOFT WINDOWS 2000/XP system Preparation Tool (imaging tool):

[0011] The Windows 2000 System Preparation Tool (Sysprep) enables administrators to prepare Windows 2000 System Images as part of an automated deployment. Sysprep provides the ability to reduce the number of images required in a deployment by supporting multiple mass storage controllers. Sysprep supports cloning of a computer system's Operating System, Software and settings to create an image that can be deployed to numerous target servers or workstations. If the target system has different capability, Plug and Play drivers are loaded when the target system is powered up.

[0012] Sysprep is designed to create an image file that can be cloned to computers having similar characteristics (type/models). An enterprise may take one image file and add support to thousands of type/models by updating the driver repository on the target system (under c:\pnp). A problem with this scenario is with the difficulty of maintenance of that driver repository.

[0013] SUMMARY OF THE INVENTION:

[0014] The present invention comprises a cloning process where specific drivers for the operating system and specific system type/model being cloned are transferred to the cloned system at build time and include access to the predetermined drivers required to deploy that system. A clone (replica) image of a computer system of the desired type and model is captured, components to be added later are removed from the captured image (the clone image). The image is saved at a server. When a client is introduced that needs to be configured, the clone image corresponding to the new client hardware characteristics is loaded into the client. In a preferred embodiment, the clone image that is to be loaded into the client has no driver software included. This makes the clone image much smaller. An install program comprising an interrogation program detects attributes of the client (or target) system. The install program creates a component list comprising information about attached devices, configuration information such as amount of memory, and machine type and model number. The component list is sent to the build server which reviews the component list in conjunction with a set of predetermined rules. The build server assembles the required components including device drivers, and component programs and transmits them to the target computer system. Only the needed drivers or programs are loaded to the target (cloned) computer and the target computer is booted with the required drivers.

[0015] Before cloning the target system, a cloning preparation tool is configured to look in a predefined location on the primary partition for the driver repository.

[0016] The user clones the operating system image to the target system.

[0017] After the operating system has been cloned, a program scans the bus of the system target system and generates a list of devices to be installed. A program then contacts the build server and requests predetermined rules for the hardware identified. This program also requests rules for the type or model detected and software to be potentially installed on the system., using the type/model information of the target computer, reads the information (from a server side database or table) about the type/model that is to be built. Included in the information obtained from the server is the particular driver information required to build that type/model as well as the common drivers that are needed on the system. The program accesses predetermined rules in order to select the required drivers and programs as well as install prerequisites and install sequences to be used.

[0018] The program then transfers the necessary driver files and programs to the target computer system partition and drivers are sent in the correct driver repository.

[0019] The target system is booted and all device drivers and applications are installed at the desired versions.

[0020] It is an object of the present invention to create a clone image without components including drivers.

[0021] It is also an object of the present invention to provide a subset of available drivers to a cloned image, the subset determined by predetermined rules and target computer hardware information, the information comprising type, model or a predefined user input parameter.

[0022] It is yet another object of the present invention to provide a program to a target client system wherein the program interrogates the target system hardware information and produces a request to a remote server for a clone image or components (such as drivers) appropriate for the target system.

[0023] The above as well as additional objectives, features, and advantages of the present invention will become apparent in the following written description.

BRIEF DESCRIPTION OF THE DRAWINGS:

FIG. 1 is a diagram depicting example components of a computer system;

Fig. 2 is a diagram depicting example components of a client-server network;

Fig. 3 is a diagram depicting example components of cloned systems;

Fig. 4 is a diagram of components of cloned systems of the present invention;

Fig. 5 is a flow chart depicting steps of cloning;

Fig. 6 is a flow chart depicting steps of cloning of the present invention; and

Fig. 7 is a flow chart depicting steps of installing drivers in a cloned system.

DESCRIPTION OF THE PREFERRED EMBODIMENTS:

[0024] Fig. 1 depicts the elements that make up a typical computer for use in presenting and maintaining an application. The computer 100 consists of a Base Computer 101 which comprises a processor 106, storage media such as a magnetic disk 107 and a high speed volatile main memory 105. An operating system and application programs 111 reside on the storage media 107 and are paged into main memory 105 as needed for computations performed by the processor 106. The Base computer may include optional peripheral devices including a video display 102, a printer or scanner 110, a keyboard 104, a pointing device (mouse) 103 and a connection 108 to a network 109. In a client environment, a user will interact with a (Graphical User Interface) GUI by use of a keyboard 104 and mouse 103 in conjunction with the display of information on the display 102 under control of an application program (application 1) 112. The client application program 112 will then interact with remote users by way of the network 109.

[0025] In Fig. 2 an example Internet system is shown. A user 210 at client 1 201 uses applications on his system. This user (user 1 210) at client 1 201 can interact with clients 2-4 202-204 by way of a client server computer 206. Applications 112 may be provided by each client 201-205 and or the client server 206 or some remote server 208 by way of the network 207. The user at client 1 201 can interact with a remote user (user 5 211) at client 5 205 by way of the Internet 207.

[0026] One way that computers interact via networks such as the Internet is by using the HyperText Transfer Protocol (HTTP) open standard designed by the World Wide Web Consortium (W3C) and standardized as Internet Engineering Task Force (IETF) RFC 2616. It is an intentionally simple and open protocol that is implemented across many heterogeneous computer systems.

[0027] One example of a self configuring computer system is taught in US Patent No. 5,668,992 "Self-Configuring Computer System" assigned to IBM. This patent is incorporated herein by reference. It describes a self configuring computer system which configures and loads appropriate software to support custom configurations without manual intervention by the user or the manufacturer. When the computer is first activated, an "interrogation" module is loaded that scans the system for components requiring specific software to operate. The interrogation module enables a startup module to select software from disk-based files including the proper version of the operating system. This patent does not discuss such functions as: creating a clone image appropriate for a group of computer systems, providing cloned images to computer systems, providing a separate driver table for finding appropriate drivers for the characteristics of a cloned system, adding drivers to a system so that the operating system's plug and play engine will detect and install the necessary drivers. It also does not address build time dynamic installation of drivers on cloned systems.

[0028] Patent Application IBM Docket No. POU920020111US1 "BUILD TIME DYNAMIC INSTALLATION OF DRIVERS ON CLONED SYSTEMS" assigned to IBM describes a cloning method providing a predefined table of components for a cloned system and is incorporated herein by reference.

[0029] In a preferred embodiment of the present invention (reference Fig. 4), a cloning computer program is provided for creating a replica (clone) computer system program image 401. A computer system 301 is prepared for a cloning operation. The preparation configures a computer system 301 that has required characteristics of target computer systems to be replicas. Target computer systems 307 312 will be the recipients of the clone image 401. A set rules 402 for building the target computer 307 is also created that preferably contains identifiers identifying desired components such as device drivers for a predefined group of computer systems compatible with the characteristics required. The rules 402 preferably contain rules for sequencing the installation of components and prerequisites as well as component version appropriate for the combination of operating system and client support requirements. The set of rules 402 includes a first identifier of a first component. Each identifier may include a component name, the location of where to get the component, the parameters needed to assist a program in retrieving the component, a component version indicator and the like.

[0030] A clone image 401 of the prepared target computer system 301 is created that includes an operating system and preferably has some or all of the initial components 306 303 deleted. The clone image 401 is transferred to a second computer system 307. Preferably, an interrogation program running at the second system verifies that the interrogation program is the correct version by communicating with the build program 406 at the build server 316. The interrogation program further detects second system 307 attributes including the type and model 325 and amount of installed memory of the second system 307. The interrogation program further detects devices attached to the second system 307. (Programs that perform interrogation of attributes and devices when a system is "booted" (power up sequence) are called wizards in operating systems such as WINDOWS 2000 from MICROSOFT CORP. Such wizards are run on built computer systems and depend on functionality of devices having been previously installed. They, further depend on having access to device drivers that have previously been loaded).

[0031] The interrogation program scans the target system, generates a list of device ids, queries the server database for matching rules, then, base on the rules, it gets code and instructions on what to do with that code. The build program 406 uses the results in association with build rules 402 to determine the components needed by the target computer 307. The components preferably comprise device drivers, configuration information, application programs and install instructions or sequences.

[0032] The components needed to complete build 403 are sent to the second system 307. The first identifier in the component list is used to locate the first component and insert it into transferred clone image 401 on the second system 307.

[0033] We will refer to the new system to be created using a common (cloning) system image 401 as the target system 307. The common (cloning) system image 401 is the cloning image and is defined for an imaginary system we call the template system. Before cloning the target system 307 the cloning preparation tool is configured to look in a predefined location on the primary partition of the target system 307 for the driver repository. The user clones the operating system 302 to the target system 307. After the operating system has been cloned a cloning program reads the information about the target system type/model from the build's dynamically created response file in the target system 307.

[0034] The cloning program transfers the drivers 403 to the primary NTFS preferably using a tool such as "Pqaccd.exe" from PowerQuest. Pqaccd.exe enables one to write to NTFS Partitions. NTFS partitions are ordinarily protected from user access and the PowerQuest tool is able to access these partitions. The image files created are saved as NTFS partitions. Pqaccd.exe is a tool/executable that permits one to edit the protected partition. The program transfers the drivers to the primary NTFS partition in the correct driver repository. FAT, FAT32, and NTFS are different partition/file system types. Different partition/file types (depending on the file system) provide better security, support for larger hard drives, and speed. The PowerQuest tool then edits the driver repository located on the primary partition (adds the driver support). The target system is booted and all device drivers are installed at the desired version, according to the definition for that model/type in the database or table.

Description of file names used:

[0035] "mtbchknt.exe" - A program written to read the type/model information about a system from an input file then perform actions based on the values in the input file. - This program does most of the work according to the present invention.

[0036] "mrf.ini" - the build input file. Generated by selections chosen by the user and type/model specific information retrieved from server.

[0037] "config2k.exe" - GUI mode program used to install type/model specific software.

[0038] "pqaccd.exe" - A third party tool developed by PowerQuest to write to NTFS partitions.

[0039] "sysprep.inf" - control file for the Microsoft Clone Preparation Tool "Sysprep"

[0040] "mtb.ini" - master type/model table. Contains the type/model specific build information for ever supported system. During the build process the information stored in this table is transferred to the build response file (mrf.ini).

[0041] "c4ebcust" - GUI mode install control program.

[0042] "IBM Standard Client Installer (ISCI)" - Build process used by clients deployed within IBM.

[0043] The technical details for a preferred implementation are as follows:

[0044] After the clone operation is complete at the target system 307, an interrogation program at the target system 307, in network communication with a build program 406 at the build server 316, checks the build server database to ensure that the latest version of the interrogation program is running. If a newer version exists, it is downloaded to the target system 307 and started, replacing the old interrogation program.

[0045] The interrogation program scans the PCI bus of the system 307 (or whatever interconnect bus that is being used by the system), and generates a device list of detected devices attached to the target system. The program also detects the type/model 325 of the system 307 and any other hardware information available 310 about the system including but not limited to amount of memory (RAM), processor speed, hard disk space and the like).

[0046] Based on the device list, the client contacts the build server 316 database. The build program 406 at the build server has access to "rules" 402 that determine what happens to the client based on the devices detected and user specific requirements if needed.

[0047] Referring to Fig. 5, the clone process begins by creating a clone image 501 of a desired system. The clone image is preferably saved at a build server so it is available via a network to target systems. The clone image could also be distributed by CDRom or other techniques known in the art. A cloning tool running at a target computer clones the image to the target computer. A program preferably comprising an interrogation program determines configuration information 503 (including ID's of attached devices as well as computer model/type and options installed information. The program in cooperation with the build server determines the needed drivers and configuration information according to the rules and downloads them to the target computer. In one embodiment predetermined tasks are performed based on instructions in the build rule. Drivers are transferred to the primary partition of the target computer system 504. When the system is rebooted with the clone image and devices 505, the drivers are installed 506.

[0048] In an example embodiment, a target system 307 has a CD-RW optical device attached. The interrogation program detects and records the CD-RW ID. The interrogation program queries the build server 316 for a "rule" for the CD-RW using the ID. It determines from the rule, what must be done. In an alternate embodiment, rule interrogation is done at the build server 316 rather than by the interrogation program. In the alternate embodiment, the build server resolves constraints and provides needed drivers and programs. An example rule would instruct the interrogation program to:

1. Download a file that updates the registry for the operating system on the target computer 307 during the first "boot" (configuration) operation.

2. Download CD-RW software to be installed via a post-build automated process.

3. Download special driver 404 to be installed for that device.

4. Apply other customizations, run other executables or download other code for that CD-RW device.

[0049] Based on the Type/Model 325 detected, the interrogation program checks for a "rule" for the target system 307 and performs customizations. An example would:

1. Detect the target system Type/Model 325 is a 2366-97U (IBM ThinkPad T30).

2. Obtain the rule 402 for the 2366-97U from the build server 316 which states a ThinkPad T30 must have CD-RW software installed (even if not detected). This might be because it is a prerequisite for other software, or because the user will soon be getting a CD-RW device but it is not yet installed.

[0050] The interrogation program, in an example embodiment, checks for application installation prerequisites. The program interrogates the build server rules 402 for all applications defined for the target configuration based on such elements as memory size (RAM), Processor type or number of processors, hard disk size, and target system type/model. In an example embodiment:

1. Interrogation program scans the build database 402 for application rules and finds a rule for MICROSOFT OFFICE XP. The rule says to install the MICROSOFT OFFICE XP if the hardware detected has at least 256 MB RAM, a P4 2.2 GHz processor and 40GB hard disk.

2. During the first OS boot, MICROSOFT OFFICE XP will be installed.

[0001] While the preferred embodiment of the invention has been illustrated and described herein, it is to be understood that the invention is not limited to autonomic computing or the precise construction herein disclosed, and the right is "reserved" to all changes and modifications coming within the scope of the invention as defined in the appended claims.